Архитектура II

Кристиян Стоименов

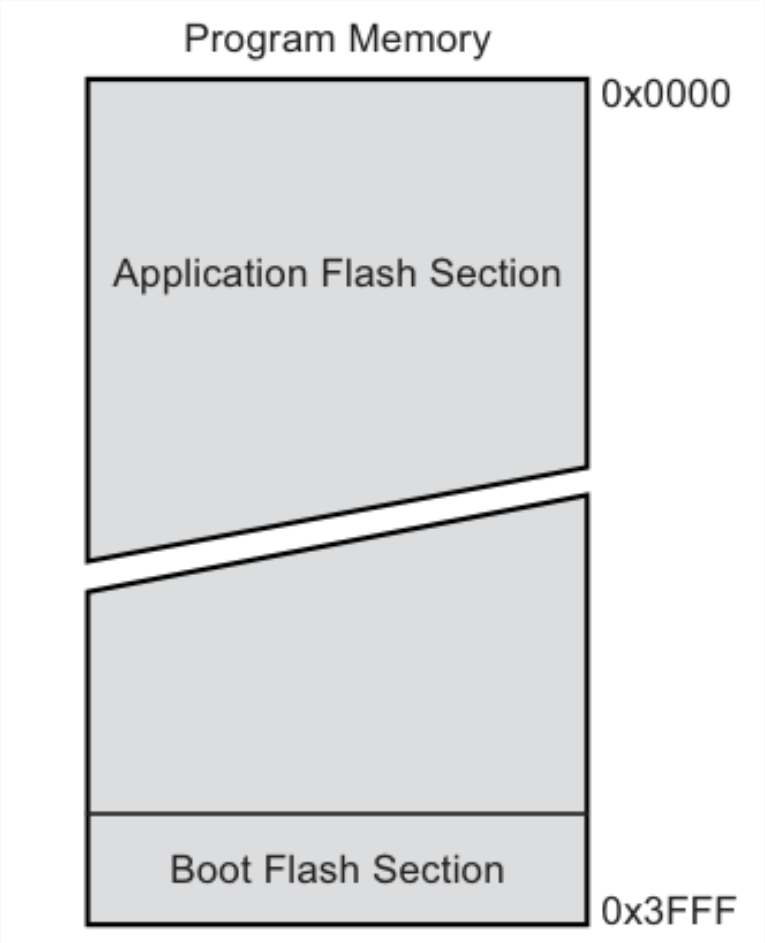
1 ноември 2023 г.

ТУЕС,
ПВМКС

Памет

The AVR architecture has two main memory spaces, the data memory and the program memory space. In addition, the ATmega328P features an EEPROM memory for data storage.

Flash


Program Memory
0x0000
Application Flash Section
Boot Flash Section
0x3FFF

Flash

- 16KB програмируема Flash памет;
- Там се съхранява програмният код;
- Всяка инструкция е по 16B;
- 10000 write/erase цикъла.

Flash

Какъв размер очакваме да има PC?

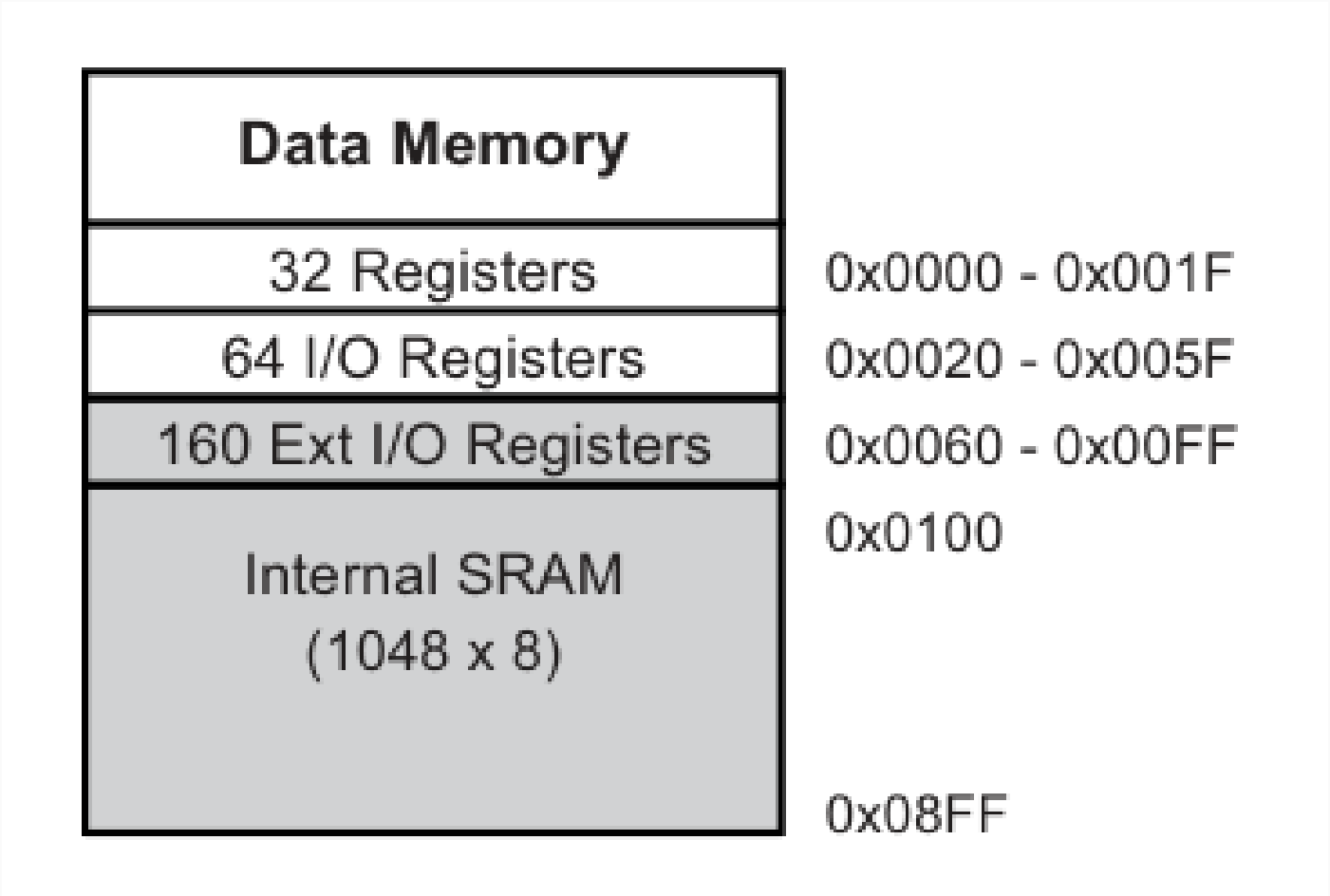
Flash

Какъв размер очакваме да има PC?

$$16KB = 2^5 KB = (2^4)^{10} B = 2^{14} B$$

$\Longrightarrow$ PC трябва да бъде поне 14-битов

SRAM


Data Memory
32 Registers
64 I/O Registers
160 Ext I/O Registers
Internal SRAM
(1048 x 8)
0x0000 - 0x001F
0x0020 - 0x005F
0x0060 - 0x00FF
0x0100
0x08FF

SRAM

- 4KB SRAM;
- Там се съхраняват данните на програмата;
- Достъпват се със специални инструкции - `IN`, `OUT`, `ST/STS/STD`, `LD/LDS/LDD`;
- `ATmega328P` поддържа пет адресни режима, за да достъпва `SRAM`.

EEPROM i

- 1KB;
- Енергонезависима, елекрически адресируема;
- Предназначена предимно за четене;
- Ограничен брой операции за писане - 100000 write/erase цикъла;
- Предимно се използва посредством серийни интерфеси - I^2C, SPI.
- Най-ефикасно се чете последователност от байтове;
- EEARL, EEDR, EECR;
- EEPROM.h;

EEPROM

EEARL - адрес, с който работим;

EEDR - при *писане* съдържа данните, които трябва да се запишат; при *четене* съдържа прочетените данни;

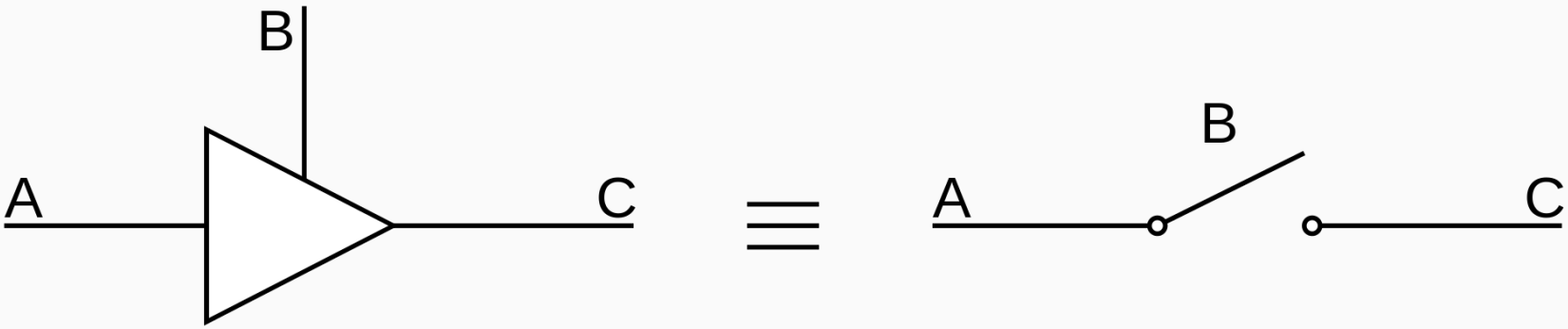
EECR - разни неща, сред които и избор на операцията.

EEPROM

```
void EEPROM_write(unsigned int uiAddress, unsigned char ucData)
{
        /* Wait for completion of previous write */
        while(EECR & (1<<EEPE))
        ;
        /* Set up address and Data Registers */
        EEAR = uiAddress;
        EEDR = ucData;
        /* Write logical one to EEMPE */
        EECR |= (1<<EEMPE);
        /* Start eeprom write by setting EEPE */
        EECR |= (1<<EEPE);
}
```

Портове

Tristate i



Tristate ii

INPUT		OUTPUT
A	B	C
0	0	Z (high impedance)
1		Z (high impedance)
0	1	0
1		1

- Три състояния - ниско, високо и високо-импедансно (*Hi-Z*;
- При *Hi-Z* изходът от буфера е практически откачен от изходната шина;
- Полезно, когато имаме няколко устройство, закачени на една и съща шина и желаем те да се редувам, когато я използват;
- Съществуват алтернативни подходи, които ще разгледаме по-нататък.

В случая с `Arduino`:

- Два извода от един и същи порт могат да се "променят" независимо;
- Всеки извод си има асоцииран *pull-up* резистор;
- Изводите могат да осигурят достатъчно ток, че да управляват LED правилно;

За всеки порт има по три стойности - `PORTx`, `DDRx` & `PINx`;

- `PORTx` - ако изводът е конфигуриран като output, то 1 в съответния бит на този регистър активира *pull-up* резистора;
- `DDRx` - "посока" на данните; при 1 - output; при - input;
- `PINx` - данните от порта.

Посредством `MCUCR` могат да се спрат всички `pull-up` резистори.

- вж. стр. 60, таблица 13-1;
- вж. стр. 61, C пример.

Литература

- *"Atmel328P Datasheets"*. URL: https://gitlab.com/tues-embedded/vmks/-/blob/master/DatasheetsAtmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf?ref_type=heads/ (дата на посещ. 28.09.2023)